# Direct Preference Optimization: Your Language Model is Secretly a Reward Model

**Rafael Rafailov**[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

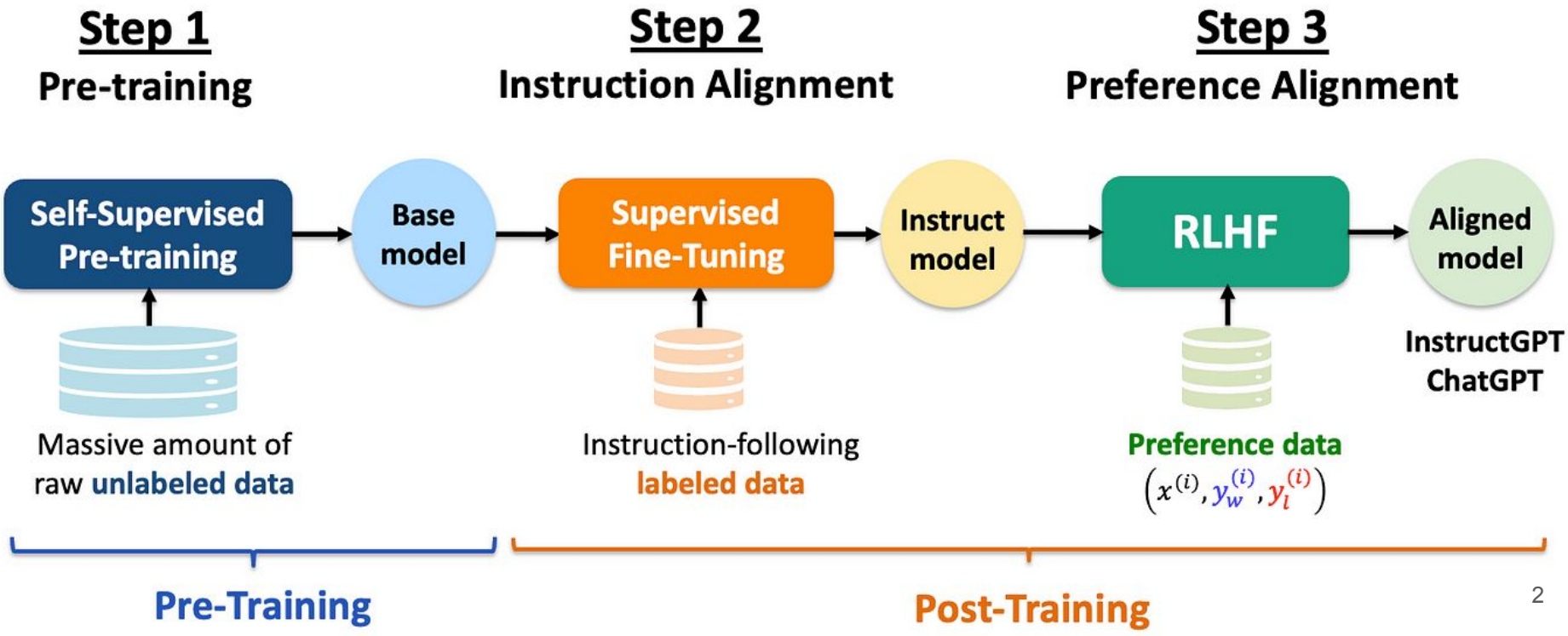
**Stefano Ermon**[†‡] **Christopher D. Manning**[†] **Chelsea Finn**[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

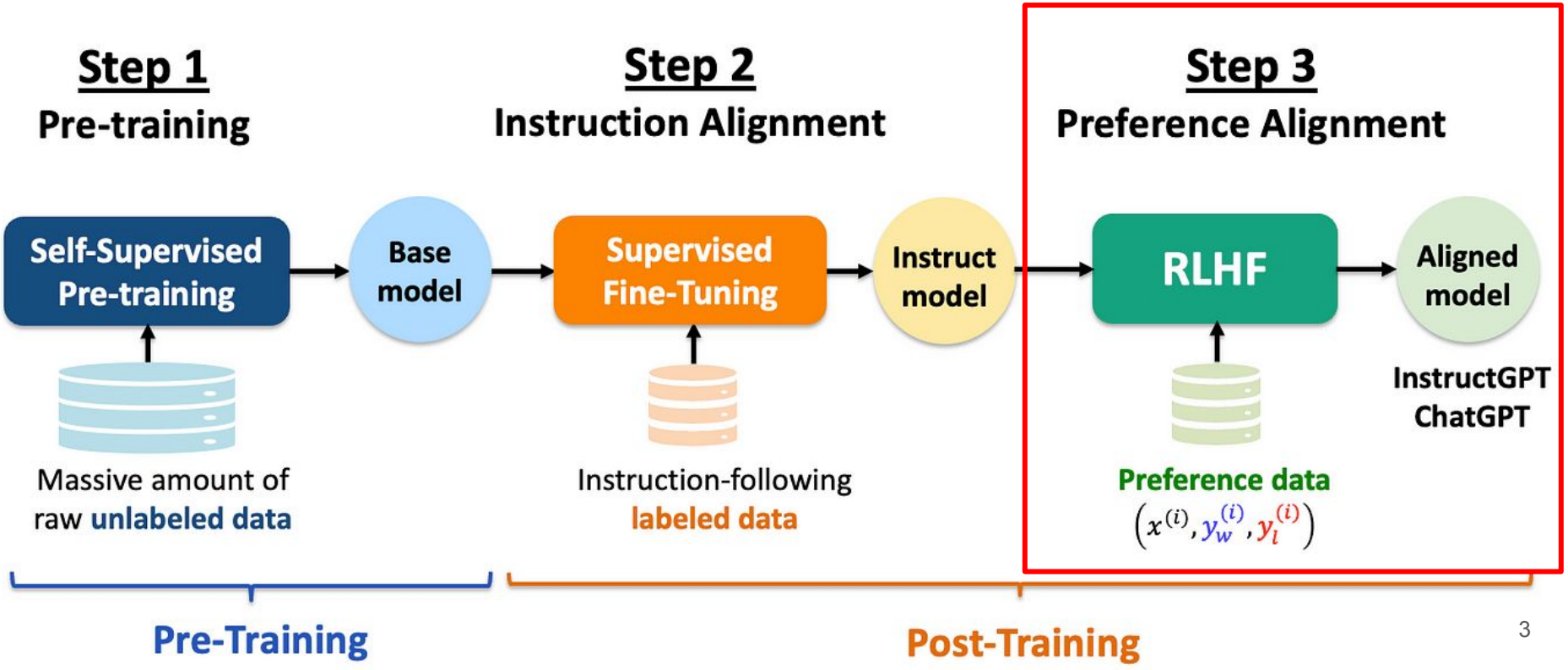
Presenters: Keane Zhang, Pranav Somu

NeurIPS 2023
Outstanding Paper Runner-up

# Language Model (LM) Training Steps



# Language Model (LM) Training Steps



in a 10-minute video, how many words are there on average?

You're giving feedback on a new version of ChatGPT.

Which response do you prefer? Responses may take a moment to load.

Response 1

The number of words in a 10-minute video depends on the speaking speed. On average:

- **Slow pace (100–130 words per minute):** 1,000–1,300 words
- **Normal pace (130–160 words per minute):** 1,300–1,600 words
- **Fast pace (160–200 words per minute):** 1,600–2,000 words

If you're working on a script or transcription, aim for around 1,500 words for a conversational tone. Would you like help calculating for a specific style?

I prefer this response

Response 2

The number of words in a 10-minute video depends on the speaking speed. On average:

- **Slow pace**: ~100 words per minute (~1,000 words in 10 minutes)
- **Average pace**: ~130–150 words per minute (~1,300–1,500 words in 10 minutes)
- **Fast pace**: ~180–200 words per minute (~1,800–2,000 words in 10 minutes)

If the video involves dense narration or rapid dialogue, it could lean toward the higher end. For conversational or casual videos, it may be closer to the average.

I prefer this response

in a 10-minute video, how many words are there on average? $x$

You're giving feedback on a new version of ChatGPT.

Which response do you prefer? Responses may take a moment to load.

**Response 1**

The number of words in a 10-minute video depends on the speaking speed. On average:

- **Slow pace (100–130 words per minute):** 1,000–1,300 words
- **Normal pace (130–160 words per minute):** 1,300–1,600 words
- **Fast pace (160–200 words per minute):** 1,600–2,000 words

If you're working on a script or transcription, aim for around 1,500 words for a conversational tone. Would you like help calculating for a specific style?

I prefer this response

$y_1$

**Response 2**

The number of words in a 10-minute video depends on the speaking speed. On average:

- **Slow pace**: ~100 words per minute (~1,000 words in 10 minutes)
- **Average pace**: ~130–150 words per minute (~1,300–1,500 words in 10 minutes)
- **Fast pace**: ~180–200 words per minute (~1,800–2,000 words in 10 minutes)

If the video involves dense narration or rapid dialogue, it could lean toward the higher end. For conversational or casual videos, it may be closer to the average.

I prefer this response

$y_2$

# Bradley-Terry Model

- The Bradley–Terry model estimates the relative preference strength of candidates from pairwise preference labels:

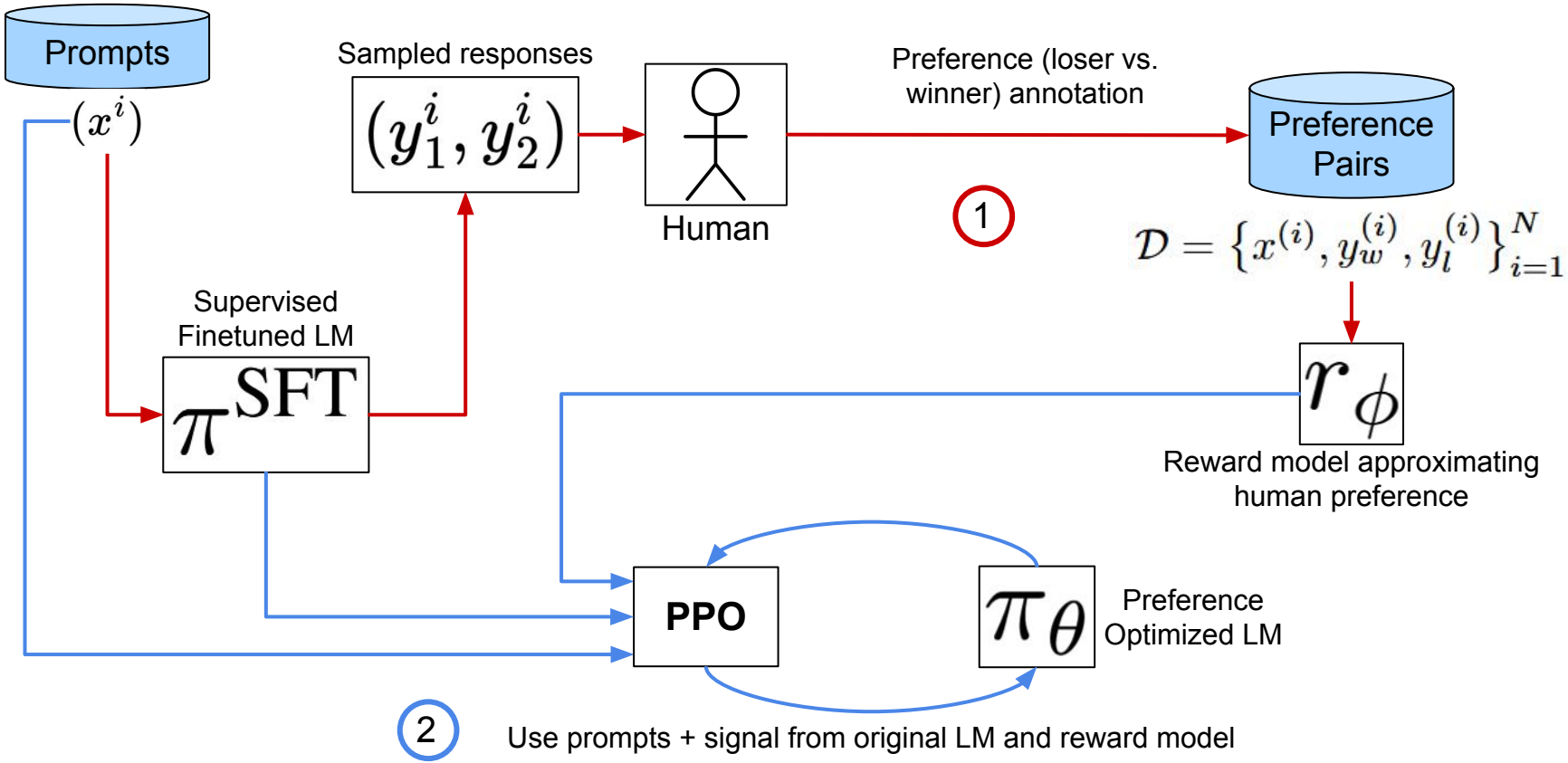
$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right) + \exp\left(r^*(x, y_2)\right)} = \sigma\left(r^*(x, y_1) - r^*(x, y_2)\right)$$

- $p^*(y_1 \succ y_2 \mid x)$ : (ground-truth) probability that $y_1$ is preferred over $y_2$ given $x$
- $\sigma$ : sigmoid function
- $r^*(\cdot, \cdot)$ : latent (ground-truth) reward function



R. A. Bradley and M. E. Terry, “Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons,” Biometrika, vol. 39, no. 3/4, pp. 324–345, 1952, doi: 10.2307/2334029.

# RLHF: Reinforcement Learning from Human Feedback



# RLHF: Training the Reward Model

1. Initialization:

$$r_\phi = (\pi^{\text{SFT}} \text{ followed by a linear layer})$$

2. Fitting:

$$\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\right]$$

Minimize as binary classification, with negative log-likelihood considering difference of reward

$\sigma$ : sigmoid activation

$\mathcal{D}$ : (Prompt, Winner, Loser) dataset

# RLHF: Tuning the LM

Objective optimized via Proximal Policy Optimization (PPO, a Deep RL method):

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(y|x)}\left[r_\phi(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x) \,||\, \pi_{\mathrm{ref}}(y \mid x)\right]$$

$\pi_{\mathrm{ref}}$: unmodified instance of $\pi^{\mathrm{SFT}}$

Iteratively update $\pi_\theta$ :

1. Maximize reward
2. (Somewhat) minimize KL-Divergence between response of the tuned model and the original model
   a. Prevent the moving out of the distribution that the reward model is trained on
   b. Prevent mode collapse and retain diversity

AKA a **constrained optimization** problem

# RLHF: Why not gradient descent?

Preference optimization objective:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}} \underbrace{_{y\sim\pi_\theta(y|x)}}\left[r_\phi(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x) \,||\, \pi_{\mathrm{ref}}(y \mid x)\right]$$

$$\pi_\theta(y|x)$$

- Sampling discrete tokens $y$ from the language model
  - Sampling is not differentiable
  - Ex. language modeling: compute cross-entropy on predicted vs true token probability distributions
  - But in our case, the backprop chain breaks regardless of the sampling method:
    - Token $y_a$ in $y$ was picked randomly or using argmax
- Therefore, we need PPO/other RL methods

# Why not PPO/RL?

- However, PPO/RL has many disadvantages:
  - Approximation error from reward model
  - Potentially unstable behavior
  - Complex implementation
  - Higher compute

- Can we remove the non-differentiable component from the objective?

# Direct Preference Optimization (DPO)

There is actually a closed form solution to the RLHF objective, assuming general non-parametric policy $\pi(\cdot|x)$ :

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}\left[r_\phi(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x) \,||\, \pi_{\mathrm{ref}}(y \mid x)\right]$$

$$= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\mathrm{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right]$$

normalization constant

$$Z(x) = \sum_y \pi_{\mathrm{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)$$

# DPO: Solution to RLHF Objective

There is actually a closed form solution to the RLHF objective, assuming general non-parametric policy $\pi(\cdot|x)$ :

$$\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)} \left[ \log \frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x) \right]$$

- To minimize the log term, we must match the numerator and denominator.
- Optimize $\pi(y|x)$ (numerator) to match the denominator.
- ∴ The denominator acts as the optimal policy.

# DPO: Solution to RLHF Objective

There is actually a closed form solution to the RLHF objective, assuming general non-parametric policy $\pi(\cdot|x)$ :

$$\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right]$$

$$\pi_r(y \mid x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$$

optimal policy

constant w.r.t. $y$

reference policy

reward

# DPO: Z(x) is intractable

$$Z(x) = \sum_{y} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

- Summing over all possible token sequences of arbitrary length
- The sum has roughly $|V|^L$ terms
  - If $|V| = 50000, L = 200, |V|^L \approx 10^{940}$
- Hard to compute! Eliminate it from the expression or estimate it (not preferable)

# DPO: Getting rid of Z(x)

Optimal policy: $$\pi_r(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

# DPO: Getting rid of Z(x)

Optimal policy: $$\pi_r(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

Rearranging to obtain an expression for $r(x, y)$, the reward function:

$$r(x, y) = \beta \log \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$$

# DPO: Getting rid of Z(x)

Bradley-Terry Model:

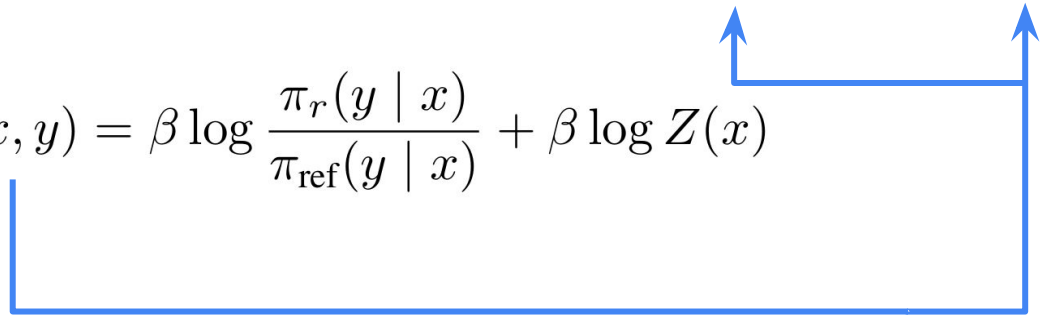
$$p^*(y_1 \succ y_2|x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right) + \exp\left(r^*(x, y_2)\right)}$$
$$= \sigma\big(r^*(x, y_1) - r^*(x, y_2)\big)$$

# DPO: Getting rid of Z(x)

Bradley-Terry Model:

$$p^*(y_1 \succ y_2|x) = \sigma(r^*(x, y_1) - r^*(x, y_2))$$

Plugging in $r(x, y) = \beta \log \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$



# DPO: Getting rid of Z(x)

Bradley-Terry Model:

$$p^*(y_1 \succ y_2|x) = \sigma(r^*(x, y_1) - r^*(x, y_2))$$

Plugging in $r(x, y) = \beta \log \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$:

$$p^*(y_1 \succ y_2|x) = \sigma\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} - \beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right)$$

- This reparameterization removes:
  - $Z(x)$: an intractable quantity
  - $r^*$: reward model

# DPO: Loss and DPO Update

The DPO Loss:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l)\sim\mathcal{D}} \left[ \log \sigma \left( \beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \right]$$

# DPO: Loss and DPO Update

The DPO Loss:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

DPO Update:

$$\nabla_\theta\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$

$$-\beta\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Bigg[\underbrace{\sigma(\hat{r}_\theta(x,y_l) - \hat{r}_\theta(x,y_w))}_{\text{higher weight when reward estimate is wrong}}\Bigg[\underbrace{\nabla_\theta\log\pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta\log\pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Bigg]\Bigg]$$

# DPO: Loss and DPO Update

DPO Update:

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$

$$-\beta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Bigg[\underbrace{\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w))}_{\text{higher weight when reward estimate is wrong}}\Bigg[\underbrace{\nabla_\theta \log \pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Bigg]\Bigg]$$

where $\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ ,

the reward implicitly defined by the language model $\pi_\theta$ and reference model $\pi_{ref}$

# DPO: Loss and DPO Update

DPO Update:

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$

$$-\beta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Big[\underbrace{\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w))}_{\text{higher weight when reward estimate is wrong}}\Big[\underbrace{\nabla_\theta \log \pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Big]\Big]$$

where $\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ ,

the reward implicitly defined by the language model $\pi_\theta$ and reference model $\pi_{ref}$

# DPO: Loss and DPO Update

DPO Update:

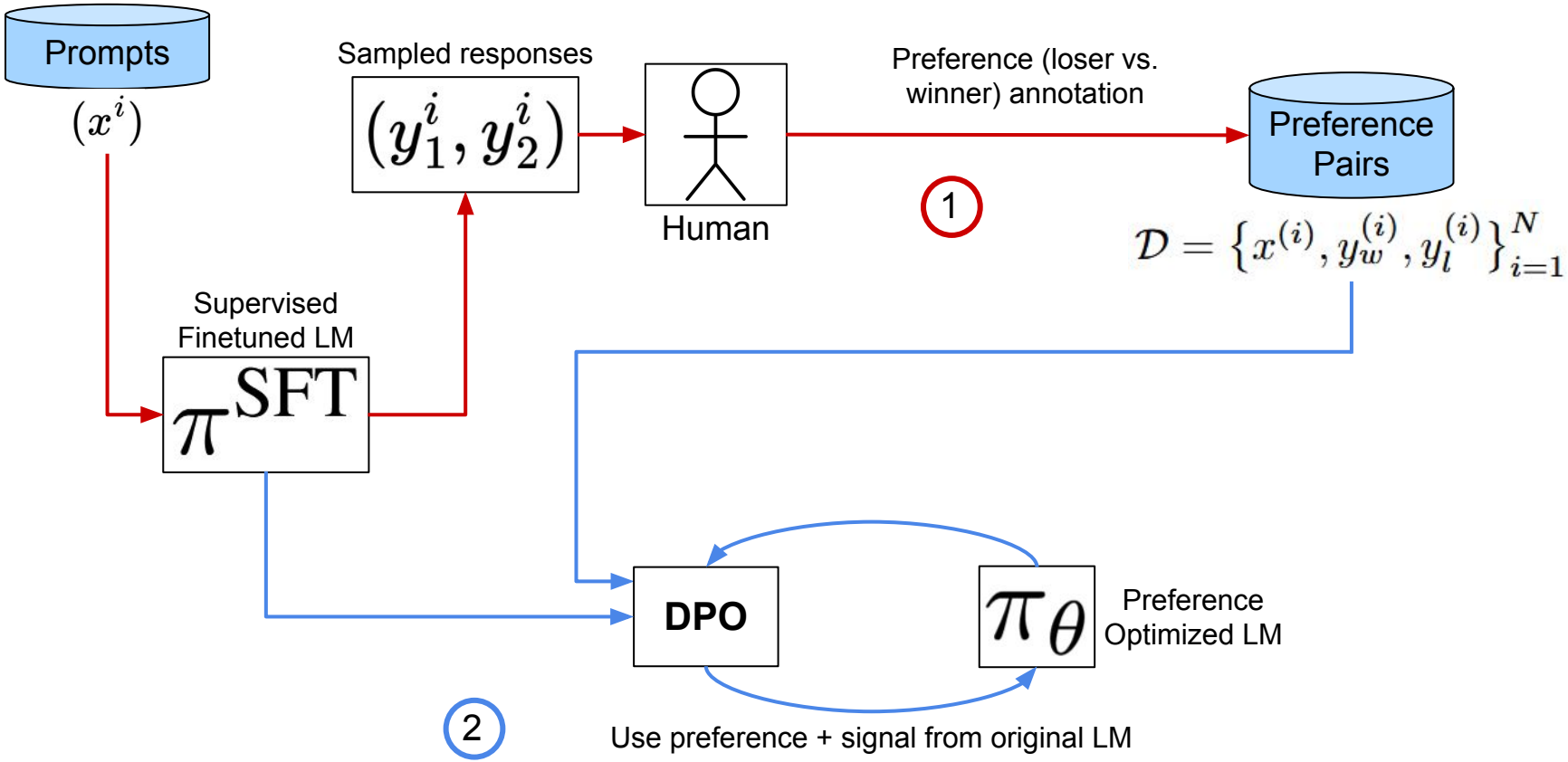
$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$

$$-\beta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Big[\underbrace{\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w))}_{\text{higher weight when reward estimate is wrong}}\Big[\underbrace{\nabla_\theta \log \pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Big]\Big]$$

where $\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ ,

the reward implicitly defined by the language model $\pi_\theta$ and reference model $\pi_{ref}$

# DPO: Direct Preference Optimization



# Comparison of RLHF and DPO

<table>
<tr><th>RLHF</th><th>DPO</th></tr>
<tr><td colspan="2">1. Pre-train and finetune a base LM</td></tr>
<tr><td colspan="2">2. Let the LM produce a pair of outputs, let humans rate</td></tr>
<tr><td>3. Train a reward model, which predicts human ratings</td><td rowspan="2">3. Train LM to assign high probability to positive examples and low probability to negative examples</td></tr>
<tr><td>4. Use RL (e.g. PPO) methods to train the LM, aiming to receive high reward from the reward model</td></tr>
</table>

DPO eliminates time, compute, and limitations of a reward model and RL methodology.

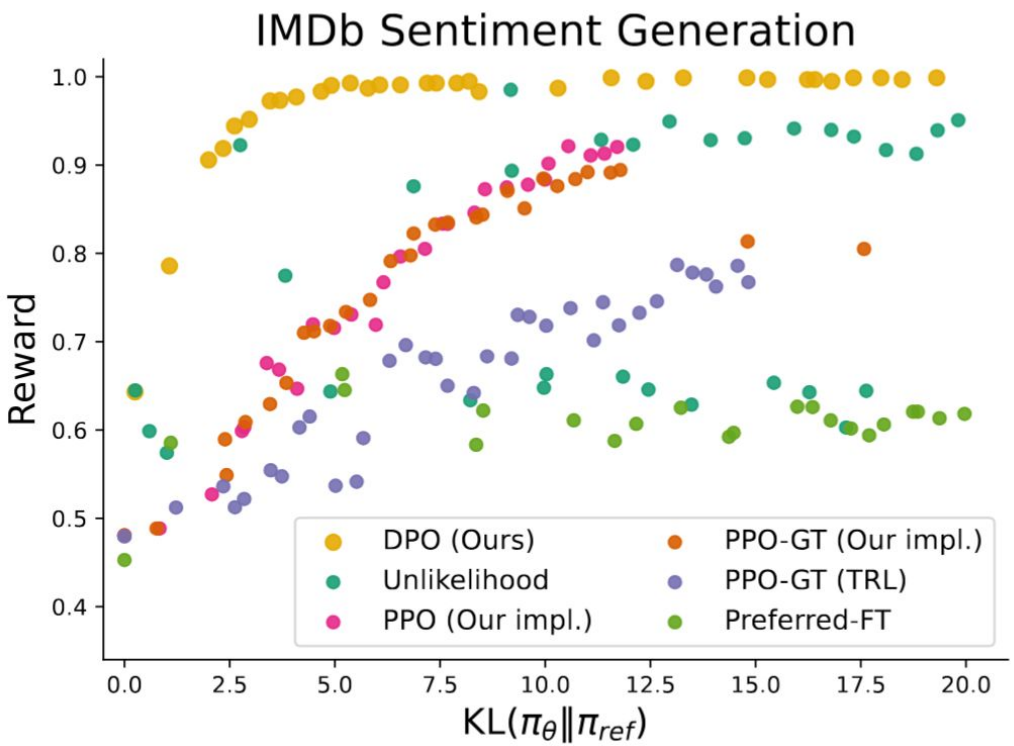
# Experiments: Alternate Methods Tested

- Preferred-FT: fine-tuned an LM with supervised learning on the chosen completion $y_w$
- Unlikelihood: simply optimizes the policy to maximize the probability assigned to $y_w$ and minimize the probability assigned to $y_l$
  - Different from DPO; it does not use the KL Constraint or dynamic weighting
- PPO: implemented in the RLHF pipeline shown earlier
- PPO-GT: uses ground truth reward function/labels (NOT a reward model that estimates reward)
- Best-of-N: samples N responses from the SFT LM and returns the response that a reward model scored best (no RL)

# Experiments - Controlled Sentiment Generation (IMDb)

- Prompt: prefix of movie review
- Output: completion with positive sentiment
- Preference pairs labeled by sentiment classifier (not human)

- GPT-4 as SFT LLM
- Y-axis: average true reward (the classifier)
- DPO produces highest reward, even with low KL

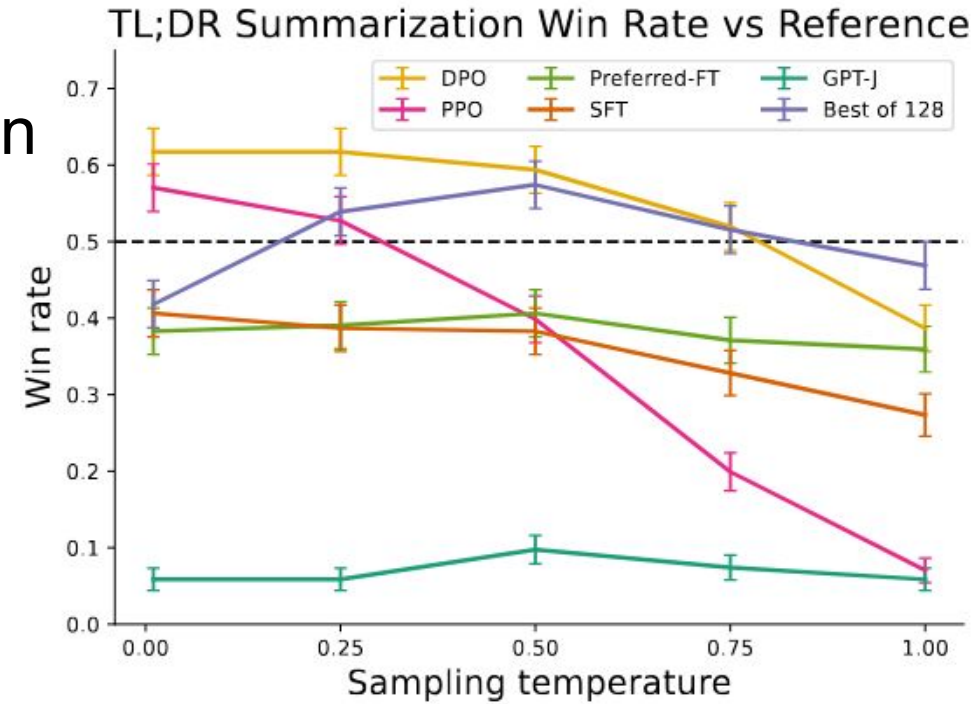


# Experiments - Summarization

- Prompt : Reddit forum post (TL;DR dataset)
- Output: summary of the post

- GPT-J LM SFT on summaries
- Y-axis: win rate against sample summaries provided in the dataset
  - GPT-4 acts as the judge, choosing between a new response vs. an existing sample

- Transfer performance to unseen CNN/DailyMail summarization task

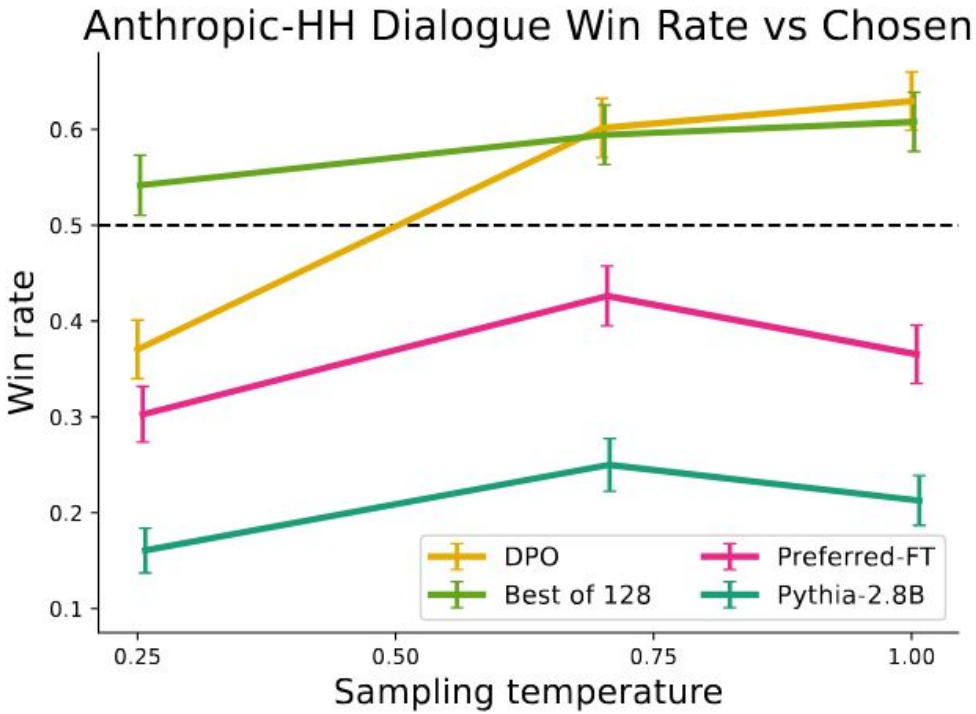



| | Win rate vs. ground truth | |
|---|---|---|
| **Alg.** | Temp 0 | Temp 0.25 |
| DPO | 0.36 | 0.31 |
| PPO | 0.26 | 0.23 |

# Experiments - Single-Turn Dialogue (Anthropic HH*)

- Prompt: user query
- Output: dialogue response

- Pythia-2.8B LM SFT on preferred dialogue (Preferred-FT)
- Y-axis: win rate against sample dialogues provided in the dataset
  - GPT-4 acts as the judge






*Anthropic Helpful and Harmless

# Conclusion

- DPO eliminates complex and potentially unstable behavior from an RL loop
  - Simple objective for constrained optimization
- An explicit reward model becomes obsolete
  - The LM itself implicitly represents reward
- SOTA performance over PPO-based methods with minimal hyperparameter tuning
- Efficiency and simplicity
  - No sampling (token output) is needed during finetuning; DPO optimizes the likelihood the model assigns to the winner vs loser samples in the preference dataset
- Robust generalization (Reddit TL;DR -> CNN/DailyMail)